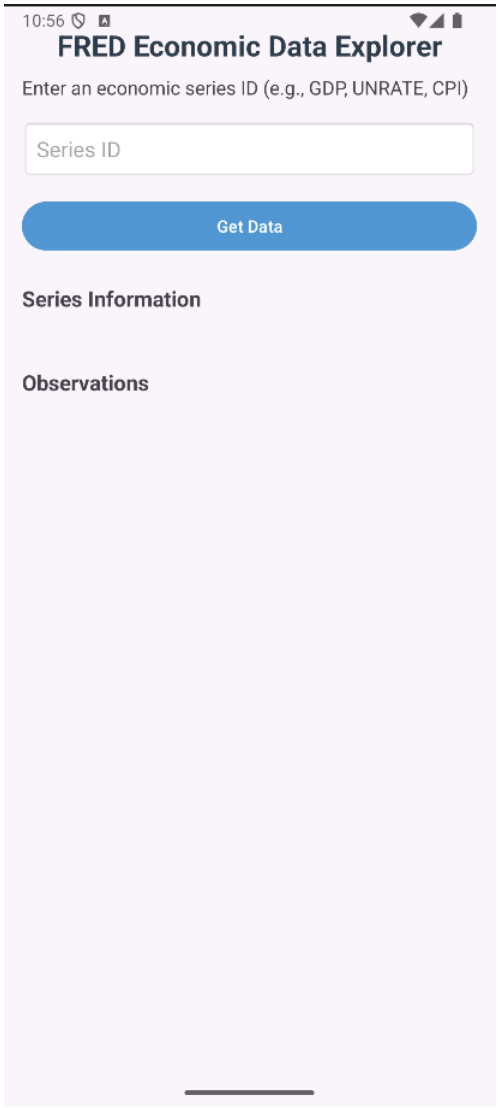
1. Android Application

Project Name in Android Studio: Project4Task2

(a) Layout with Multiple Views
Views Implemented:

- EditText (User input for FRED series ID, e.g., "GDP")
- Button (Triggers data fetch)
- TextView (Displays metadata and observations)

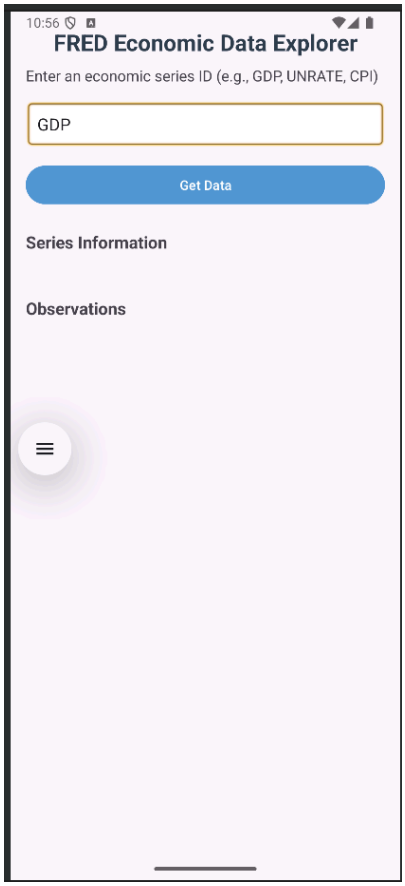


Layout File: activity_main.xml

```java
<EditText
    android:id="@+id/seriesIdEditText"
    android:hint="Series"/>
<Button
    android:id="@+id/getDataButton"
    android:text="Get Data"/>
<TextView
    android:id="@+id/seriesInfoTextView"/>
<TextView
    android:id="@+id/observationsTextView"/>
```

(b) User Input Handling
Users enter a FRED series ID (e.g., "GDP") and submit via button click:

```java
getDataButton.setOnClickListener(v -> {
    String seriesId = seriesIdEditText.getText().toString().trim();
    if (!seriesId.isEmpty()) {
        new FetchFredDataTask().execute(seriesId); // Async HTTP request
    } else { Toast.makeText(MainActivity.this, "Please enter a series ID",
Toast.LENGTH_SHORT).show();
            }
});
```



(c) HTTP Request to Web Service
A GET request is sent to your backend API:

```java
URL url = new URL(API_URL + "?seriesId=" + URLEncoder.encode(seriesId,
"UTF-8"));
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
connection.setRequestMethod("GET");
```

(d) JSON Response Parsing
The backend returns JSON, parsed as follows:

```java
// Parse JSON response
JSONObject jsonResponse = new JSONObject(result);
```

```
// Check for error in response
if (jsonResponse.has("error")) {
    seriesInfoTextView.setText("Error: " + jsonResponse.getString("error"));
    return;}

// Display series information
if (jsonResponse.has("seriesInfo")) {
    JSONObject seriesInfo = jsonResponse.getJSONObject("seriesInfo");

StringBuilder infoText = new StringBuilder();
infoText.append("Series: ").append(seriesInfo.getString("title")).append("\n");
infoText.append("Frequency:").append(seriesInfo.getString("frequency")).append(
"\n");
infoText.append("Units: ").append(seriesInfo.getString("units")).append("\n");

seriesInfoTextView.setText(infoText.toString());
     }
```

(e) Display Results
Data is rendered in TextViews:

```
Java
// Display series information
if (jsonResponse.has("seriesInfo")) {
                    JSONObject seriesInfo =
jsonResponse.getJSONObject("seriesInfo");
                    StringBuilder infoText = new StringBuilder();
                    infoText.append("Series:
").append(seriesInfo.getString("title")).append("\n");
                    infoText.append("Frequency:
").append(seriesInfo.getString("frequency")).append("\n");
                    infoText.append("Units:
").append(seriesInfo.getString("units")).append("\n");
                    seriesInfoTextView.setText(infoText.toString());
                }

// Display observations
if (jsonResponse.has("observations")) {
       JSONArray observations = jsonResponse.getJSONArray("observations");
       StringBuilder obsText = new StringBuilder("Recent Data Points:\n\n");

       for (int i = 0; i < observations.length(); i++) {
```

```
            JSONObject obs = observations.getJSONObject(i);
            obsText.append("Date: ").append(obs.getString("date"))
                                    .append(", Value:")
                                    .append(obs.getString("value"))
                                    .append("\n");
                    }
         observationsTextView.setText(obsText.toString());
        }
```

2. Web Application (Deployed to GitHub Codespaces)

API Endpoint: `https://solid-goggles-v6gx6vxgxwvpf6qqj-8080.app.github.dev/api`

(a) HttpServlet API Implementation

Servlet: ApiServlet.java

```java
@WebServlet("/api")
public class ApiServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse
response) {
        String seriesId = request.getParameter("seriesId");
        // Call FRED API and return JSON
    }
}
```

(b) Business Logic

1. Accepts seriesId from the Android app.
2. Calls FRED API:

```java
String fredUrl =
"https://api.stlouisfed.org/fred/series/observations?series_id="
                + seriesId + "&api_key=YOUR_FRED_KEY";
```

3. Parses FRED's JSON response and extracts key data.

3. Log useful information - Itemize what information you log and why you chose it.

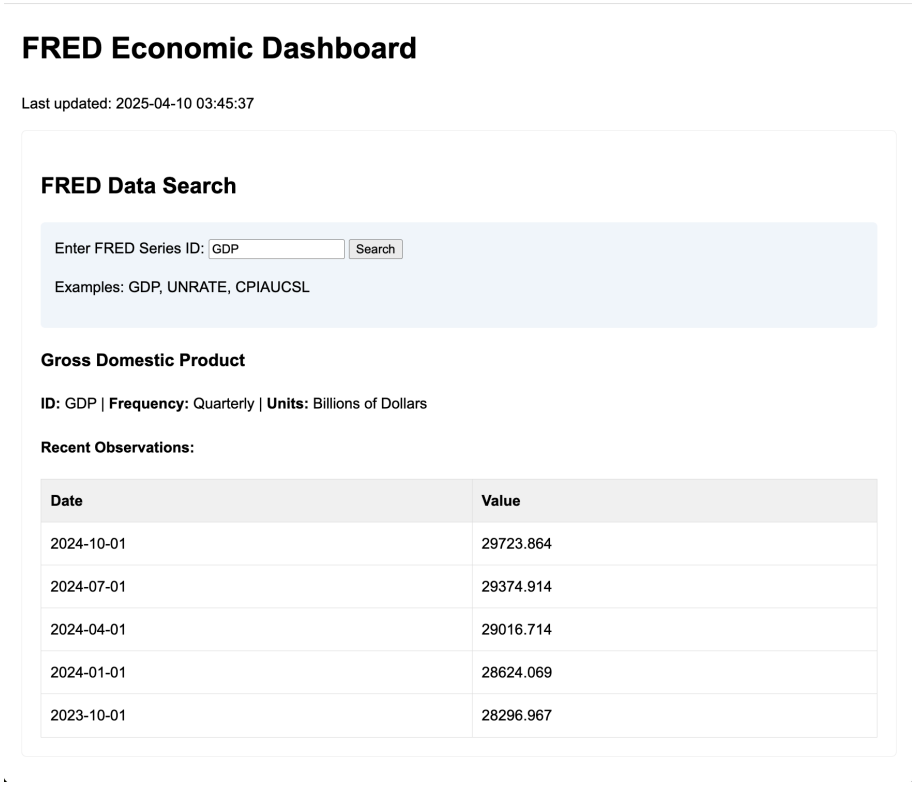
Logged Field	**Reason for Choosing**
`timestamp`	Track when events occurred
`type`	Distinguish request vs response
`query`	Record which economic indicator was requested
`userAgent`	Identify client devices/browsers
`responseTime`	Performance monitoring (responses only)
`responseData`	Verify API outputs (truncated to 500 chars)

4. Database Connection

MongoDB Atlas Connection

```java
private static final String CONNECTION_STRING =
"mongodb+srv://yanxuand:XGUn2JAPjDJFfthF@cluster0.plzd6.mongodb.net/?retryWrite
s=true&w=majority&appName=Cluster0";
private static final String DATABASE_NAME = "project4_demo";
private static final String LOG_COLLECTION = "app_logs";
private static final String DATA_COLLECTION = "economic_indicators";
```

5. Display operations analytics and full logs on a web-based dashboard - Provide a screenshot.



FRED Economic Dashboard

Last updated: 2025-04-10 03:45:37

FRED Data Search

Enter FRED Series ID: GDP Search

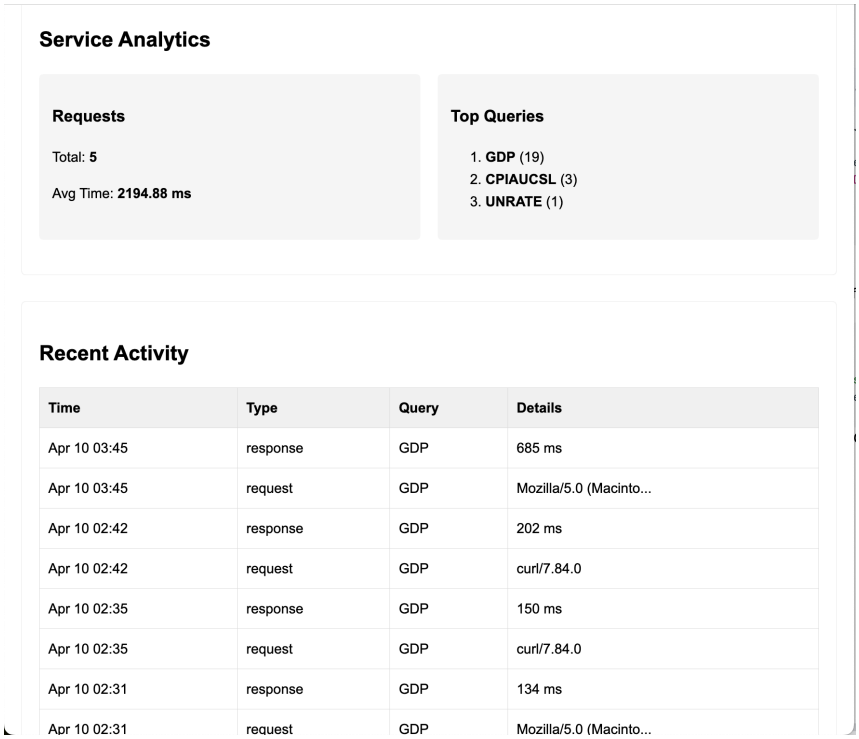
Examples: GDP, UNRATE, CPIAUCSL

Gross Domestic Product

ID: GDP | **Frequency:** Quarterly | **Units:** Billions of Dollars

Recent Observations:

Date	Value
2024-10-01	29723.864
2024-07-01	29374.914
2024-04-01	29016.714
2024-01-01	28624.069
2023-10-01	28296.967



Service Analytics

Requests

Total: **5**

Avg Time: **2194.88 ms**

Top Queries

1. **GDP** (19)
2. **CPIAUCSL** (3)
3. **UNRATE** (1)

Recent Activity

Time	Type	Query	Details
Apr 10 03:45	response	GDP	685 ms
Apr 10 03:45	request	GDP	Mozilla/5.0 (Macinto...
Apr 10 02:42	response	GDP	202 ms
Apr 10 02:42	request	GDP	curl/7.84.0
Apr 10 02:35	response	GDP	150 ms
Apr 10 02:35	request	GDP	curl/7.84.0
Apr 10 02:31	response	GDP	134 ms
Apr 10 02:31	request	GDP	Mozilla/5.0 (Macinto...